Volume 3, No. 1, January 2012
Journal of Global Research in Computer Science
RESEARCH PAPER
Available Online at www.jgrcs.info
ISSN-2229-371X

APPLICATION BASED SELECTION TECHNIQUES FOR WEB SERVICES MECHANISM

Deshraj Ahirwar*[1] Sandeep Gupta, Upendra and Neelesh Kori
[*1]Department of C.S.E, Samrat Ashok Technological Institute, Vidisha (MP)
deshrajahirwar.sati@gmail.com
[2]Department of C.S.E, Samrat Ashok Technological Institute, Vidisha (MP)
sgsandy30@gmail.com
[3]Department of C.S.E, Samrat Ashok Technological Institute
upendra.chaurasiya@gmail.com
[4]Department of C.S.E, Samrat Ashok Technological Institute
neelesh.kori01@gmail.com

Abstract- It has become much more difficult to access relevant information from the Web With the explosive growth of information available on the World Wide Web. One of the promising approaches is web usage mining, which mines web logs for user models and recommendations. In the domain of Web Services, it is not uncommon to find redundant services that provide functionalities to the clients. Services with the same functionality can be clustered into a group of redundant services. Respectively, if a service offers different functionalities, it belongs to more than one group. Having various Web Services that are able to handle the client's request suggests the necessity of a mechanism that selects the most appropriate Web Service at a given moment of time. In this paper presents an approach, Repository Based Web Services Selection, for dynamic service selection based on virtualization on the server side. It helps managing redundant services in a transparent manner as well as allows adding services to the system at run-time. In addition, the model assures a level of security since the consumers do not have direct access to the Web Services.

Keywords- Web Service, QoS, Service Selection.

INTRODUCTION

Web Services can encapsulate a specific task or can be designed as a composition of other services, representing a complex aggregation. The Web Services conceptual model describes the process of discovery, request and response. Discovery is the process of finding the service that provides the functionality that is required. A request provides the input to the service. The response yields the output from the service. Service providers describe their Web Services and advertise them in a universal registry called Universal Description Discovery and Integration (UDDI). This enables service requestors to search the registry and find services. UDDI allows for the creation of registries that are accessible over the web. IBM, Microsoft, and Oracle all have public UDDI servers running for commercial purposes. There are also many organizations developing third party servers for users to establish their private UDDI servers. The UDDI client offers two approaches to access the UDDI server: either through a standalone application providing an easy-to-use interface for developers; or through a software library working with the WS consumer or provider. UDDI Browser is an open-source UDDI client following the first approach. A developer can use the application to browse, search, and even change information in the UDDI server [1] and [3].

Web Services can be ranked by the Quality of Service (QoS) they offer. QoS is a means to enable selection and filter out unqualified providers. QoS can be seen as an aggregated measure of generic criteria such as availability, reliability, failure rate, trust and reputation, response time, price, and network load and domain specific features .The reasoning mechanism is responsible for the selection of a Web Service at a particular moment of time. In order to distinguish one service from another using the specified criteria, this unit requires a set of instructions that help evaluate each component and choose the most appropriate one respectively. A set of instructions can be seen as a selection technique.XML Web services enable the exchange of data and the remote invocation of application logic using XML messaging to move data through firewalls and between heterogeneous systems. Although remote access of data and application logic is not a new concept, but doing so in a loosely coupled fashion is. Hence it poses new challenges. In Web Services, the interface hides the implementation details of the service, allowing it to be used independently of the hardware and software platform on which it is implemented and also of the programming language in which it is written. This allows and encourages Web Services based applications to be loosely coupled, component oriented, cross technology implementations [4].

In this paper we proposed the web service selection technique. The purpose of web service selection is to select optimal web service for a particular task. When dynamic discovery is used in Web Services, it is common that the result of the discovery contains more than one provider. Unlike the file sharing P2P system in which a file download can be split into many small tasks running in multiple peers, a service invocation occurs between a provider and a consumer. The Web Service consumer must pick only one from all candidate providers to perform the invocation. Even for a composite Web Service consisting of many atomic Web Services, the selection issue still needs to be addressed when there are multiple providers available for an atomic service. In order to make a distinction between the services which provide the same functionality, selection criteria should be used. They help evaluate the Web Services within a group and choose the component that matches the needs and the preferences of the consumers, while taking into account the abilities of the providers.

BACKGROUND

Web Service:

The definition of a Web service is given as: "any process that can be integrated into external systems through valid XML documents over Internet protocols". This definition outlines the general idea of Web services are built for unlike services in general, Web services are based on specifications for data transfer, method invocation and publishing. This is often misunderstood and when a Web service is mentioned it sometimes refers to a general service provided on the Web, like the weather forecast on a Web page for example. The weather forecast is a service and provides its functionality for a variety of users but unless it comprises an interface to communicate with other applications via SOAP, it is not a Web service by definition. Web services can be seen as software components with an interface to communicate with other software components. They have certain functionality that is available through a special kind of Remote Procedure Call. In fact they even evolved from traditional Remote Procedure Calls. The difference lies in the interface and the method for transportation. Furthermore Web services cannot be viewed or used with an ordinary browser. They require a unified form of messaging embedded in a XML document. This communication architecture contains three subcomponents.

Consumer: This denotes the entity utilizing the Web service. This is another application in most cases.

Transport: It defines the means for the communication the Consumer uses while interacting with a service.

Provider: The service provider.
In order to keep the whole system truly platform-independent, transport in both direction uses XML. This includes the description of an operation to execute and the data payload as well. Although transportation is not restricted to a specific protocol or method, HTTP became the most popular way to pass on XML documents between Web services [2] and [6].

Simple Object Access Protocol (SOAP):

SOAP, the Simple Object Access Protocol was developed to enable a communication between Web services. It was designed as a lightweight protocol for exchange of information in a decentralized, distributed environment. SOAP is an extensible, text-based framework for enabling communication between diverse parties that have no prior knowledge of each other. This is the requirement a transport protocol for Web services has to fulfill. SOAP specifies a mechanism to perform remote procedure calls and therefore removes the requirement that two systems must run on the same platform or be written in the same programming language. SOAP also defines data encoding rules, called base level encodings. It is important to note that these Section 5 encodings are not mandatory in any way, so clients and servers are free to use different conventions for encoding data as long as they agree on format. All this is done in the context of a standardized message format.

The primary part of this message has a MIME type of text/xml and contains the SOAP envelope which is an XML document. The envelope consists of a an optional header which may target the nodes that perform intermediate processing, and a mandatory body which is intended for the final recipient of the message. This way a firewall can be adjusted to filter SOAP Messages with an inappropriate header for example. The Header may also hold digital signatures for a request contained in the body. The body contains the serialized payload. For a request this is the method argument where the surrounding XML tag must have the same name as the called method. The response body contains the return value if it exists. Data types are not delineated in the SOAP envelope explicitly so the type of a result parameter cannot be discovered just by looking at the SOAP message [4] and [5].

Using Web Services:

To finally use a Web service, several steps have to be performed. The order of the events, followed by a description of how to execute each step.

Locating the Web service: This can either be done by browsing a public UDDI registry or by means of an existing WSDL document. It is possible to build a private UDDI registry as well. Private registries are easier to maintain due to their size but it can be hard to discover the UDDI registry's position. Sometimes, a company's main Web page is linked to WSDL documents, too.

Creating the SOAP Message: This is done by the development tool in most cases. Tools like Weblogic Workshop from BEA or Web service Development Kit from Microsoft will create valid SOAP messages for the methods described in the WSDL document or UDDI registry.

Transmission: Another advantage of message transport via HTTP is the service providers firewall setting. If the firewall permits Port 80 (HTTP POST/GET) connections, a SOAP message is able to pass through as well. If the firewall is unable to filter and process SOAP requests on the other hand, it leaves the system vulnerable to attackers who use the Web service's functionality for a potential attack.

Parsing the SOAP message : This is done by the provider's Application Server. The parser decides if the request is valid and decides which procedure to call.

Processing: The service provider calls all necessary procedures, or even other Web services, to complete the requested task.

Return the Result: The result is wrapped in a SOAP reply and returned to the requestor where the client application can parse the message and evaluate the included data [6] and [7].

Related Works:

Researchers have proposed various approaches for dynamic web service selection. Maximilien and Singh proposed a multi-agent based architecture to select the best service according to the consumers' preferences. Maximilien and Singh describe a system in which proxy agents gather information on services, and also interact with other proxy agents to maximize their information and the conceptual model they use to interact with the services is detailed elsewhere. The proxy agents lie between the service consumer and the service providers. The agents contact a service broker, which contains information about all known services, as well as ratings about its observed QoS. From there, the information is combined with its own historical usage, and the combined knowledge is used to select a service, though the authors do not detail how. The agencies contain data about the interactions between the clients and the services which is used during the Web Services selection process. In his work, trust and reputation are taken into account during the decision process. Their approach divide the QoS attributes into objective and subjective. The former include QoS features such as availability, reliability, and response time [3] and [8] .

In recent proposed a features, in their proposed approach as well but their major selection criteria is based on the QoS based service selection. They have considered three quality criteria namely execution time, execution duration and reputation for the selection. In addition, execution price, duration, transactions support, compensation and penalty rate are the other criteria. The authors of suggest an open, fair, and dynamic framework that evaluates the QoS of the available Web Services by using clients' feedback and monitoring.

The reasoning mechanism is responsible for the selection of a Web Service at a particular moment of time. In order to distinguish one service from another using the specified criteria, this unit requires a set of instructions that help evaluate each component and choose the most appropriate one respectively. A set of instructions can be seen as a selection technique. The major components of a reasoning mechanism are criteria, model, and selection technique. The model collects information about the participants of the client-server interaction as well as represents it as aggregated measures. Different selection techniques can implement various business logics in order to make a decision.

The reasoning mechanism in the approach proposed by Liu, Ngu, and Zeng computes the QoS of the Web Services, ranks them, and selects the most appropriate one. To perform the selection, the QoS registry in their system takes in data collected from the clients, stores it in a matrix of web service data in which each row represents a web service and each column a QoS parameter, and then performs a number of computations on the data, such as normalization. Clients can then access the registry, and are given a service based on the parameters that the client prefers. The bottleneck of the approach is the dependency on the consumers to give regular feedback about their past experience with the Web Services. The Success of this model is based on the clients or the end users and their will to provide the necessary feedback on QoS [9] and [10].

PROPOSED TECHNIQUES

In the domain of Web Services, it is not uncommon to find redundant services that provide functionalities to the clients. According to the Web Services conceptual model, the client receives a list of services from the Universal Description Discovery and Integration (UDDI), selects one, and starts an interaction with the service to process the request. In previous, service selection is an important process and various techniques have been proposed. In our approach for dynamic service selection and invocation is introduced, which has the following advantages in comparison with previous approaches:

a. It provides location and replication transparency of the Web Services;
b. It hides the system's complexity from the clients;
c. It provides a transparent service selection from the client's point of view;
d. It assures a level of security, since the clients do not have direct access to the Web Services.

The development of the proposed model in a real-world application and the evaluation of such a system is a complex procedure. It requires resources in terms of machines that run a set of experiments and time that should be devoted to each experiment setup, run, and analysis. In addition, it is very likely that the results of the experiments depend on the particular machine specifications and environment settings, such as web server, communication style (for example, Tomcat and Axis framework, if the system is implemented in Java). Fluctuations, due to network, memory, CPU, caching, and garbage collection, might appear as well. All this could reflect on the correct analysis of the obtained data and the validity of the conclusions. Furthermore, such an implementation is based on particular standards, protocols, and programming languages.

Proposed Repository Based-Web Services Selection:

We propose a technique for dynamic selection of Web Services which will also handle the problem of redundant Web Services. In this work, we introduce a model with a Web Service repository, as shown in figure 1 will act as an independent unit possessing a definite functionality. This repository will be used to redirect the client's request. This will also provide a level of security since it will not be allowed to invoke directly by the clients. This technique will prevent unauthorized access to the real services. This provision will also help to hide the systems complexity from the clients.

The repository will perform three functions namely, storing, collecting and reasoning. In storing operation a QoS feedback report is generated by the client and is saved in the repository. The QoS feedback report provides a historical reference for the consumer to assess the provider. Each provider only keeps the feedback information relevant to it. The collecting operation retrieves all necessary data from providers for the reasoning operation. The reasoning operation manages to select the best service provider for the consumer according to the collected data. Consider an example where clients needs the services (S1,S2) as in figure1, it sends a request .The collecting, storing and reasoning mechanism interacts with the web services to find the most appropriate of the services and the results are stored in the repository for future reference. Web Services here interacts with the reasoning mechanism to find out the appropriate services. Once the service is selected, the request is forwarded to it. Finally, when the result is generated, it is passed to the repository which sends it back to the client.

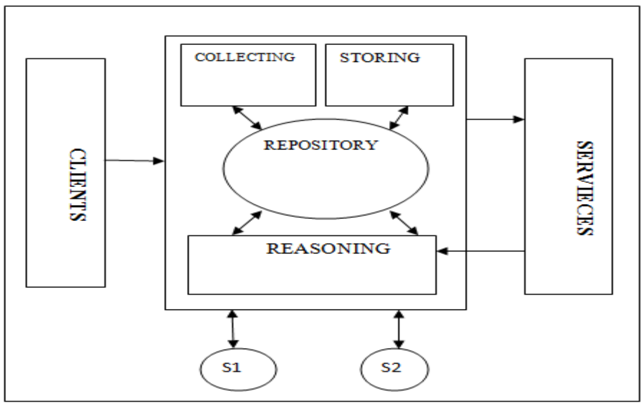

Figure1: Repository based Web Service Selection

Algorithm: Selection of Service:

This algorithm shows the necessary steps to choose a service and get the maximum quality results.

a. For finding a service for a specified task, perform a search on service descriptions.
b. Arrange all discovered services by their signature parameter and discard all other services.
c. Get the desired Service Parameters.
d. Collect the services result and order by their utility.
e. If no results are found, let the client reconsider the constraints, go to step 2.

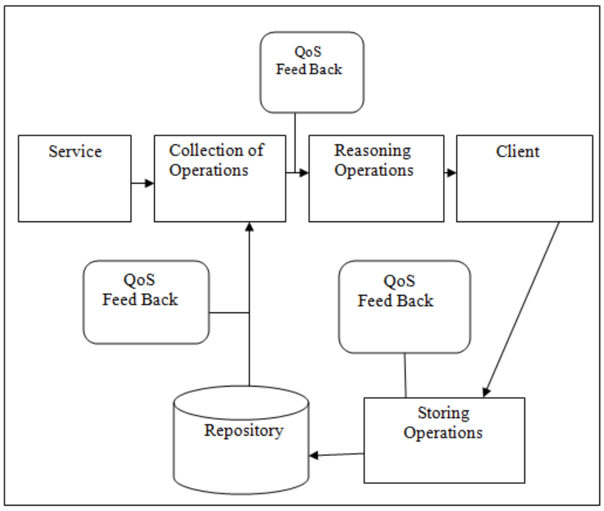

Figure 2: Selection Process

This algorithm provides an approach for selecting a specified service. Services, not matching the profile are discarded on the fly .It also helps in taking an alternative services.

Design of the Proposed Architecture:

Now we provide an overview of the design of the simulation model that is used to evaluate the Repository Based Web Services Selection approach. The architecture, proposed here consists of three main components - Clients, Repository Layer, and Web Services, forming the key compound objects of the high-level view of the simulation design. Object LG (load generator) generates clients' requests (entities); object represents the behavior of the model, and each object WS corresponds to a Web Service. Object LS (load sink) is used as the end point of the entity flow. It accepts the incoming from the Repository Based Web Services Selection entities and disposes of them. Component Manager is not included, since the settings of the reasoning mechanism are defined before the simulation runs. There is a Virtual Web Service corresponding to each group of redundant services with particular functionality. The reasoning mechanism is presented by a queue and a server.

Evaluation:

Now we evaluate of the proposed Repository Based Web Service Selection approach. Firstly, a feasibility check is done to determine whether it is possible to build such architecture. Secondly, the behavior of the Web Services is observed and analyzed in different environments. Finally, to observe the behavior of the prototype with different selection techniques, the simulation method is used, since the study of the optimal strategies is difficult in a real environment where there are many uncontrollable parameters. The evaluation process carried out by analyzing the Based Web Service Selection prototype in order to test whether or not the architecture is a feasible technique for managing redundant Web Services on the server side in a dynamic and transparent manner. A simplified prototype of the proposed architecture is developed, in order to observe if the model is an applicable approach for dynamic selection of redundant Web Services. The dynamic service selection is realized by reflection that allows us to obtain information at run-time about methods, constructors, and instance fields of classes, as well as to invoke them dynamically.

The reasoning mechanism is based on a random selection technique that does not require any information about the services. Since the decision is done in a random manner, there is no need for selection criteria, neither the data about the services must be collected and aggregated by the model. The model of the reasoning mechanism contains only the WSDL descriptions of the redundant Web Services.

RESULTS ANALYSIS

In our experiment show the results of the developed framework are as follows:

a. The model has a feasible technique for dynamic service selection on the server side. The layer is able to manage the redundant services in a transparent manner. All the necessary information, which should be available to the client, is the service description of the Web Services. The model hides the reasoning details during the decision-making process. From the consumer's point of view, the prototype is the real Web Service that handles the request.
b. The scalability of the system that implements the described layer is expected to be the same as the scalability of a system that does not consist of redundant components. The decentralization of the Web Services assures that there is no single point of control and respectively of failure. There is a separate component that represents and manages each group of services and does not influence the proper work of the whole system.

c. The architecture can be used as a layer that assures a level of security. The Web Services are called by the virtual layer and are never invoked directly by the clients. This technique can prevent unauthorized users from having access to the real services.
d. It is possible for the response time of the proposed architecture to increase due to the reasoning mechanism. This is an expected result since the decision is taken at run-time. Furthermore, it implies a trade-off between the appropriate service selection and the execution time of the system.

The obtained data shows that the execution times of the simulation runs are higher for the load balancing technique compared to the fastest service selection but lower compared to the random selection technique. In terms of Web Services overloading, both the fastest and the load balancing techniques present similar results.

CONCLUSION AND FUTURE WORKS

In this paper we proposed an approach for dynamic service selection and, which has the following advantages in comparison with previous approaches:

a. It hides the system's complexity from the clients.
b. It provides a transparent service selection from the client's point of view.
c. It assures a level of security, since the clients do not have direct access to the Web Services.

In future, other technology that can be applied in the Repository based system is Semantic Web technology. By describing the data in a machine-understandable manner and creating semantics of QoS criteria, the decision-making process would be based on more features as well as their relationships would be represented in a better and more flexible way.

REFERENCES

[1]. Li CHEN, Zi-lin SONG, Ying ZHANG, Zhuang MIAO, "A Method for Context-aware Web Services Selection", International Journal of Advancements in Computing Technology Volume 3, Number 7, August 2011, pp 291-298.

[2]. Nabil Keskes , Ahmed Lehireche, Abdellatif Rahmoun, www.ccsenet.org/cis Computer and Information Science Vol. 4, No. 3; May 2011, pp 138-150.

[3]. Hela Limam and Jalel Akaichi, "MANAGING AND QUERYING WEB SERVICES COMMUNITIES: A SURVEY", International Journal of Database Management Systems (IJDMS), Vol.3, No.1, February 2011, pp 93-108.

[4]. S.Susila, "Agent based discovery of web service to enhance the quality of web service selection", IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.2, February 2011, pp 159-164.

[5]. R. Dinesh Kumar and Dr.G. Zayaraz, "A QOS AWARE QUANTITATIVE WEB SERVICE SELECTION MODEL", International Journal on Computer Science and Engineering (IJCSE), Vol. 3 No. 4 Apr 2011, pp 1534-1538.

[6]. G. Vadivelou, E. IIavarasan, R. Manoharan, P. Praveen, "A QoS Based Web Service Selection Through Delegation", International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011, pp 1-9.

[7]. Furkh Zeshan and Radziah Mohamad, " Semantic Web Service Composition Approaches: Overview and Limitations", International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(3): 640-651 The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085), pp 640-651.

[8]. Jeberson R. Retna Raj, Sasipraba T., "Web Service Selection Based on QoS Constraints", IEEE, 2010.

[9]. Yu T., Lin K. J., "Service selection algorithms for Web services with end-to-end Qos constraints", Proceeding of Information Systems and E-Business Management, 2005.

[10]. Tran VuongXuan, Tsuji Hidekazu, "QoS based ranking for Web Services: Fuzzy Approach", International Conference on Next Generation Web Services Practices, 2008.